Discrete Applied Mathematics 159 (2011) 2266–2268

Contents lists available at SciVerse ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Note

A revised algorithm for searching for all defective edges in a graph☆

Ting Chen *

College of Statistics and Mathematics, Zhejiang Gongshang University, Hangzhou, PR China

ARTICLE INFO

Article history:
Received 14 December 2010
Received in revised form 13 July 2011
Accepted 1 August 2011
Available online 1 September 2011

Keywords:
Group testing
Graph testing
Competitive algorithm

ABSTRACT

We consider the following generalization of the classical group testing problem. Given a graph $G = (V, E)$, which contains d defective edges, we want to identify all defective edges in G by testing whether an induced subgraph contains a defective edge or not. Recently, Hwang gave a competitive algorithm to identify all defective edges in a graph with d unknown. We will show an obvious mistake in the algorithm and propose a revised algorithm to solve the problem of searching for all defective edges in a graph.

1. Introduction

The edge searching problem on graphs is an extension of the classical group testing problem. Assume that we are given a graph $G = (V, E)$ with vertex set V and edge set E. Let $G(S)$ denote the subgraph of G induced by the set S of vertices. Our task is to identify a subset $D \subseteq E$ of *defective edges* with a minimum number of *edge tests*, where an edge test takes an arbitrary subset $S \subseteq V$ and asks whether the subgraph $G(S)$ contains a defective edge.

Chang and Hwang [2] first cast the problem of identifying two defective vertices in a complete bipartite graph. This problem can be treated as a special group testing problem of searching for a single edge on graphs. Aigner [1] was the first who consciously introduced the edge testing problem for a general graph, thus bringing the "graph" into focus.

Let $|D| = d$ and denote by $M(G, d)$ the number of edge tests necessary, in the worst case, to find the d defective edges in G. Aigner [1] conjectured $M(G, 1) \leq \lceil \log_2 |E| \rceil + c$, where c is a constant. Damaschke [3] proved the conjecture with $c = 1$. Triesch [7] generalized this result to hypergraphs of rank r by proving $M(G, 1) \leq \lceil \log_2 |E| \rceil + r - 1$.

Johann [6] made a breakthrough for the $d > 1$ case by proving $M(G, d) \leq d\left(\left\lceil \log_2 \frac{|E|}{d} \right\rceil + 7\right)$ and proving a conjecture of Du and Hwang [4] that $M(G, d) \leq d\left(\left\lceil \log_2 \frac{|E|}{d} \right\rceil + c\right)$ for some constant c.

All the above results assumed that d is known. For the case that d is unknown, Hwang [5] gave a competitive algorithm to identify all defective edges in a graph requiring $d(\lceil \log_2 |E| \rceil + 4)$ tests. In this paper, we will show that there exists an obvious mistake in the algorithm and provide a revised algorithm for searching for all defective edges in a graph.

2. The competitive algorithm for graphs

In [5], Hwang gave a competitive algorithm to find all d defective edges in a graph $G = (V, E)$. We introduce the halving procedure as a subroutine of the algorithm. For a set S of n elements, the halving procedure tests a subset T of $\lceil \frac{n}{2} \rceil$ elements. If the outcome is positive, iterate the procedure on T; if negative, iterate the procedure on $S \setminus T$.

☆ This research was partially supported by the National Natural Science Foundation of China under Grant Number 10801121 and the Zhejiang Provincial Natural Science Foundation of China No. Y6100126.

* Fax: +86 571 28877222.
E-mail address: steafting@sina.com.

0166-218X/$ – see front matter
doi:10.1016/j.dam.2011.08.004

We also introduce the TJ-procedure in the algorithm. Construct a vertex cover C of E which contains c vertices, by first taking a vertex v_1 with maximum degree, then a vertex v_2 of maximum degree after v_1 and all edges incident to it are deleted, and so on. Then we test a subset $V \setminus \{v_1, \ldots, v_k\}$ for some $k < c$. If negative, we iterate the same procedure on $\{v_1, \ldots, v_k\}$. If positive, we test a smaller subset $V \setminus \{v_1, \ldots, v_l\}$ with $l > k$. The testing in this manner ends when we identify a vertex v_i such that $V \setminus \{v_1, \ldots, v_{i-1}\}$ is positive but $V \setminus \{v_1, \ldots, v_i\}$ is negative. Since v_i must be a vertex of a defective edge, we can identify a defective edge (v_i, u) with $u \in V \setminus \{v_1, \ldots, v_i\}$ by the halving procedure. Triesch and Johann proved that this process can be used to identify a single defective edge in G in $\lceil \log_2 |E| \rceil + 1$ tests even though G has many defective edges.

The following competitive algorithm to find all d defective edges in a graph $G = (V, E)$ was given by Hwang.

Algorithm.

The partition stage:

Step 1: Set $V_1 = V$, $V_2 = \cdots = V_d = \emptyset$, $I = \emptyset$ (I is the set of identified defective edges).
Step 2: Test V_1. If positive, then
- Use the TJ-procedure to identify a positive edge (v, u) where $v \in C$.
- Use the join subroutine to assign v to some V_i $(i > 1)$.
- Set $V_1 = V_1 \setminus \{v\}$, $V_i = V_i \cup \{v\}$ and $I = I \cup \{(v, u)\}$. If $|V_1| \geq 2$, go back to Step 2.

Step 3: If one of the V_j $(j > 1)$ is nonempty, we enter the search stage.
Step 4: Stop with no defective edge identified.

The join subroutine:

Suppose v is the vertex to be assigned.

Step 1: Set $i = 2$.
Step 2:
- If $(v, u) \in I$ for some $u \in V_i$, set $V_i' = \{u \in V_i : (v, u) \notin I\}$.
- Test $\{v\} \cup V_i'$. If positive, use the halving procedure to identify a defective edge (v, u).
- Set $I = I \cup \{(v, u)\}$, $i = i + 1$ and go back to Step 2.

Step 3: Add v to V_i.

The search stage:

Suppose the partition stage yields nonempty $V_1, \ldots, V_m$ for some $m \geq 2$.

Step 1: Set $j = 2$.
Step 2: For each vertex $v \in V_j$, let $V(v) = \{u \in \bigcup_{i=1}^{j-1} V_i : (v, u) \in E \setminus I\}$. Test $\{v\} \cup V(v)$. If negative, go to the next v. If positive, use the halving procedure (with v attached to every test) to identify a defective edge (v, u). Set $V(v) = V(v) \setminus \{u\}$. If $V(v) \neq \emptyset$, go back to Step 2. If $V(v) = \emptyset$, go to the next v.
Step 3: Set $j = j + 1$. If $j \leq m$, go back to Step 2.
Step 4: Stop.

Now we show that the search stage of the algorithm may not be correct. In Step 2 of the search stage, for each vertex $v \in V_j$, since $V(v) = \{u \in \bigcup_{i=1}^{j-1} V_i : (v, u) \in E \setminus I\}$, the edge set of $V(v)$ may contain a defective edge which is already identified (for the reason that $V_i \cup V_j$ $(i < j)$ may contain an identified defective edge). Thus, a future test on the vertices of $\{v\} \cup V(v)$ may encounter an identified defective edge e and the procedure of Step 2 might identify e repeatedly.

For example, we are given a graph $G = (V, E)$ with the vertex set $V = \{1, 2, 3, 4, 5, 6\}$. In this algorithm, suppose the partition stage yields $V_1 = \{1, 2, 3\}$, $V_2 = \{4, 5\}$, $V_3 = \{6\}$ and $I = \{(1, 4), (2, 5), (3, 6), (4, 6)\}$. In the search stage, for the vertex $\{6\} \in V_3$, $V(\{6\}) = \{1, 2, 5\}$ where $\{(1, 6), (2, 6), (5, 6)\} \subset E$. Then the future test on the vertices of $\{6\} \cup V(\{6\})$ will encounter the identified defective edge $(2, 5) \in I$.

We revise the search stage of the algorithm as follows.

The search stage:

Step 1: Set $i = 1$ and $j = 2$.
Step 2: Choose a vertex $v \in V_j$.
Step 3: Let $V_i(v) = \{u \in V_i : (v, u) \in E \setminus I\}$.
If $V_i(v) = \emptyset$, go to the next v and go back to Step 3.
If $V_i(v) \neq \emptyset$, test $\{v\} \cup V_i(v)$. If positive, use the halving procedure (with v attached to every test) to identify a defective edge (v, u). Set $I = I \cup \{(v, u)\}$ and go back to Step 3. If negative, go to the next v and go back to Step 3. Do this for every $v \in V_j$. Then go to Step 4.
Step 4: Set $j = j + 1$. If $j \leq m$, go back to Step 2.
Step 5: Set $i = i + 1$. If $i < m$, set $j = i + 1$ and go back to Step 2.
Step 6: Stop.

Lemma 1. *The algorithm with the revised search stage identifies all defective edges of G.*

Proof. The revised algorithm consists of a partition stage and a new search stage. In the partition stage, V is partitioned into $V_1, V_2, \ldots, V_m$ such that no V_i contains a defective edge. The join subroutine assigns a vertex v to a set V_i such that $i = \min\{j \geq 1 \mid$ There is no defective edge in $\{v\} \cup V_j(v)\}$. Some defective edges are identified along the way with its two vertices assigned to different V_i and V_j.

In the search stage, we still need to identify all remaining defective edges whose vertices spread into different subsets. We test every single vertex v mixed with every single subset of partitioning. After a test with a positive outcome on the vertices of $\{v\} \cup V_i(v)$, the procedure of the search stage will identify a new defective edge by the halving procedure as long as such an edge exists. Since it examines all unidentified edges between different subsets V_i and V_j, the algorithm with the new search stage identifies all defective edges in G. □

Based on the above discussion, we have:

Theorem 1. *Let $G(V, E)$ be an arbitrary graph which contains d defective edges. Then the revised algorithm identifies all defective edges of G with at most $d\lceil \log_2 |E| \rceil + d^2 + 3d + 1$ tests.*

Proof. Each defective edge is identified by either the TJ-procedure in $\lceil \log_2 |E| \rceil + 1$ tests, or the halving procedure in $\lceil \log_2 |E| \rceil$ tests. Thus the number of tests consumed in each identification procedure is bounded by $\lceil \log_2 |E| \rceil + 2$, including the possible positive test initiating the identification process. Then the d defective edges cost a total number of at most $d(\lceil \log_2 |E| \rceil + 2)$ tests.

We count negative tests occurred elsewhere except those in the TJ-procedure or the halving procedure. The partition stage stops with a negative test on the vertex set V_1. Each join-subroutine ends with a negative test to assign the vertex v. Since each v to be assigned corresponds to a distinct defective edge in D, the partition stage costs at most $d + 1$ negative tests.

Since each vertex v in $\bigcup_{j=2}^{m} V_j$ corresponds to a distinct defective edge, there are at most d of them. For each fixed $i (1 \leq i \leq m - 1)$ in the search stage, each such v starts a testing process and ends with a negative test. Then at most $(m - 1)d$ negative tests occur in the search stage (not counting the negative tests in the halving procedure).

Since $m - 1 \leq d$, the total number of tests in the algorithm is at most $d(\lceil \log_2 |E| \rceil + 2) + (d + 1) + d^2 = d\lceil \log_2 |E| \rceil + d^2 + 3d + 1$. □

Acknowledgments

We thank the reviewers for their efforts to improve the readability of this paper.

References

[1] M. Aigner, Combinatorial Search, in: Wiley-Teubner Series in Computer Science, Wiley, New York, 1988.
[2] G.J. Chang, F.K. Hwang, A group testing problem on two disjoint sets, SIAM Journal on Algebraic and Discrete Methods 2 (1981) 35–38.
[3] P. Damaschke, A tight upper bound for group testing in graphs, Discrete Applied Mathematics 48 (1994) 101–109.
[4] D.Z. Du, F.K. Hwang, Combinatorial Group Testing and its Applications, World Scientific, Singapore, 1993.
[5] F.K. Hwang, A competitive algorithm to find all defective edges in a graph, Discrete Applied Mathematics 148 (2005) 273–277.
[6] P. Johann, A group testing problem for graphs with several defective edges, Discrete Applied Mathematics 117 (2002) 99–108.
[7] E. Triesh, A group testing problem for hypergraphs of bounded rank, Discrete Applied Mathematics 66 (1996) 185–188.